ADVANCED OPERATING SYSTEMS 2015

USB in a microkernel based operating system

Mierswa, Daniel ✉ daniel.b.mierswa@student.hs-rm.de
Tkocz, Daniel ✉ daniel.tkocz42@gmail.com

Agenda

- Microkernels (history and architecture)
- USB (hardware and protocol specifics)
- Challenges providing USB in microkernel based OS
- Design proposal
  - Scheduling
  - Access control
- Review and conclusions
- Open discussion

Mierswa, Daniel ✉ daniel.b.mierswa@student.hs-rm.de
Tkocz, Daniel ✉ daniel.tkocz42@gmail.com

History of microkernels

- 1980: UNIX was widespread and adopted, but monolithic aspect was seen as a problem
- BSD as a reference for "much code in kernel space"
- 1985: Mach as BSD replacement (1st gen.)
  - Memory management
  - Early stages of IPC
- Mid 90s: Reboot: L4 (Liedtke)
  - 20 times faster than 1st gen.
- Still basis for state of the art development

Mierswa, Daniel ✉ daniel.b.mierswa@student.hs-rm.de
Tkocz, Daniel ✉ daniel.tkocz42@gmail.com

Architecture of microkernels (1)

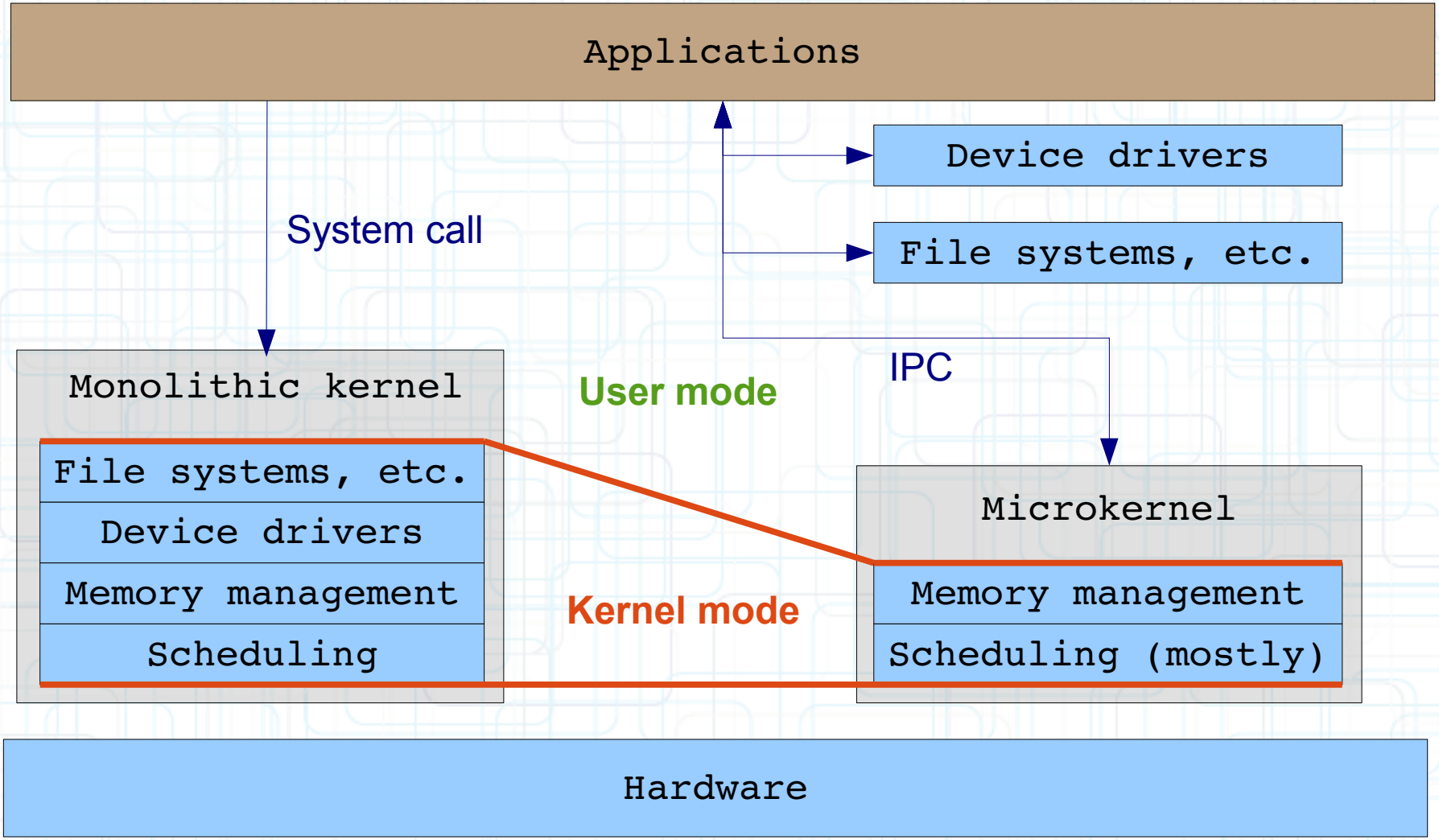
- Monolithic kernel: Hardware, driver API, etc. in kernel space
  - One big binary
  - Provide access via system calls (mode switch)
- Microkernel: Minimal code footprint in kernel space, other functionality in user space
  - Separate tasks on separate layers
  - Few system calls for message passing (context switch)
- Modular monolithic kernels: Newer Linux
  - Dynamic (un)loading of kernel modules

Mierswa, Daniel ✉ daniel.b.mierswa@student.hs-rm.de
Tkocz, Daniel ✉ daniel.tkocz42@gmail.com

Architecture of microkernels (2)

- Advantages of microkernels:
  - Kernel can continue working if services fail
  - Very modular and extensible by design
  - Separation of concern by design
- Advantages of monolithic kernels:
  - Less overhead due to fewer context switches
  - Less code due to fewer indirection (pure monolithic)
  - Smaller when compiled (pure monolithic)

Architecture of microkernels (3)



Mierswa, Daniel ✉ daniel.b.mierswa@student.hs-rm.de
Tkocz, Daniel ✉ daniel.tkocz42@gmail.com

USB Hardware

- 4 Pins (+5V, Ground, D+, D-)
- 0.5 A, 5 V
- Different Connectors (A, B, Mini, Micro, ...)
- Up to 480 Mbit/s

Mierswa, Daniel ✉ daniel.b.mierswa@student.hs-rm.de
Tkocz, Daniel ✉ daniel.tkocz42@gmail.com

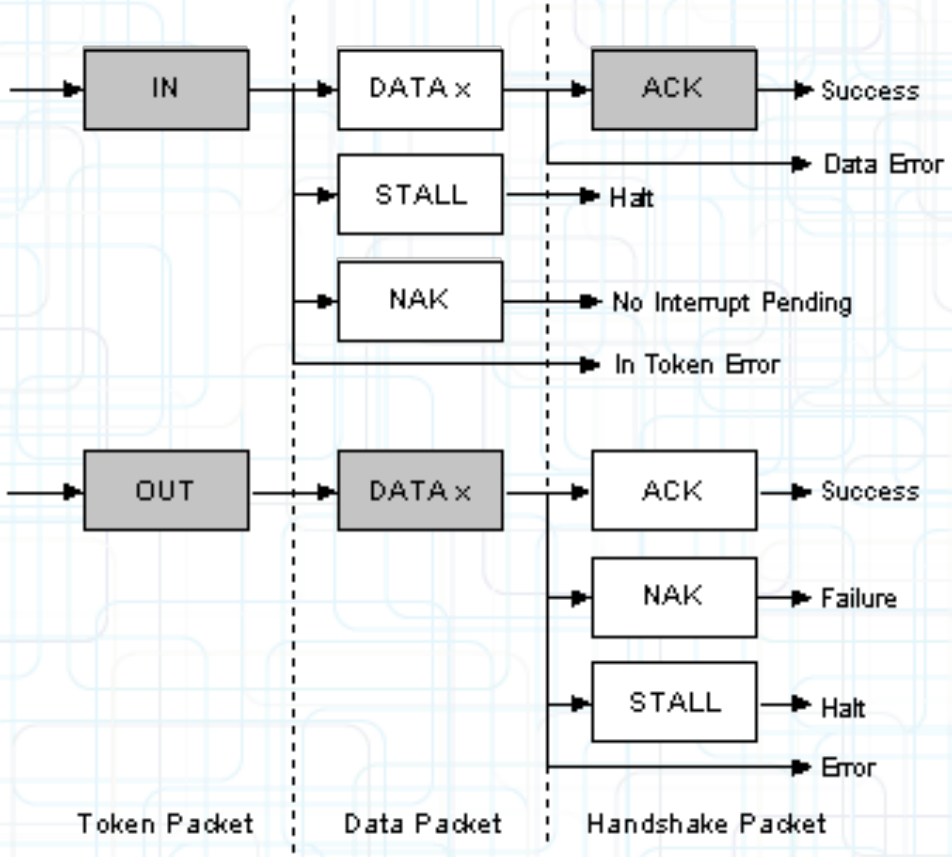
USB Protocol (1)

- Host / Client
- Device Descriptors
- Downward compatible
- Control Endpoint
- Bulk Endpoint
- Isochronous Endpoint
- Interrupt Endpoint

Mierswa, Daniel ✉ daniel.b.mierswa@student.hs-rm.de
Tkocz, Daniel ✉ daniel.tkocz42@gmail.com

USB Protocol (2)

- Interrupt Transfer Example
- Token
- Data
- Handshake



Mierswa, Daniel ✉ daniel.b.mierswa@student.hs-rm.de
Tkocz, Daniel ✉ daniel.tkocz42@gmail.com

USB Packet

- Sync
- PID (Packet ID)
- ADDR (Device)
- ENDP (Endpoint)
- DATA
- CRC (Integrity)
- EOP (End of Packet)

Mierswa, Daniel ✉ daniel.b.mierswa@student.hs-rm.de
Tkocz, Daniel ✉ daniel.tkocz42@gmail.com

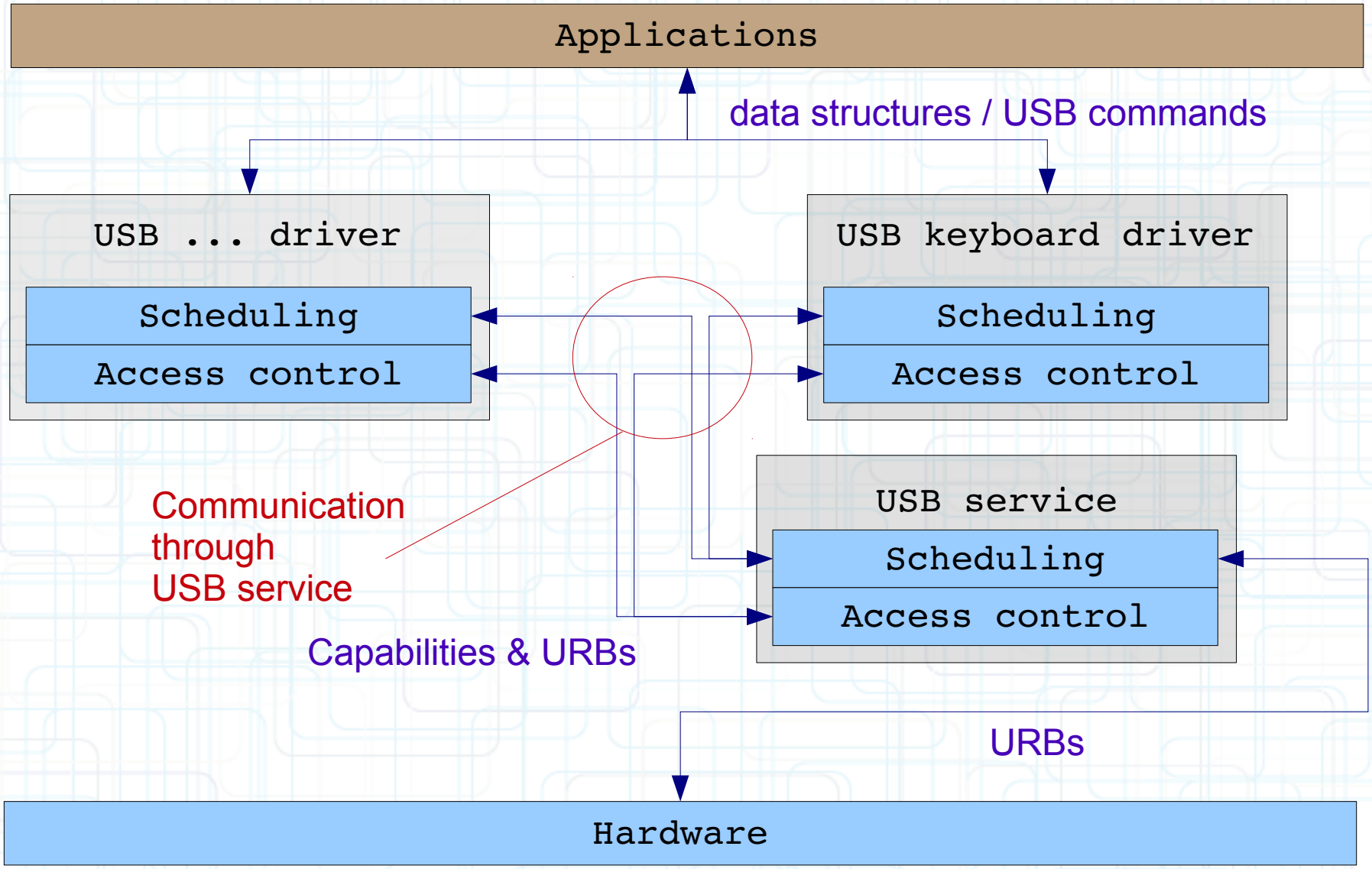
Challenges providing USB in microkernel based OS

- Main problems to solve
  - USB is a shared protocol (Bus)
  - Host controller becomes a shared resource
- Access restrictions for drivers and applications
- Central scheduling when sending/receiving URB (USB Request Blocks)

Ideas

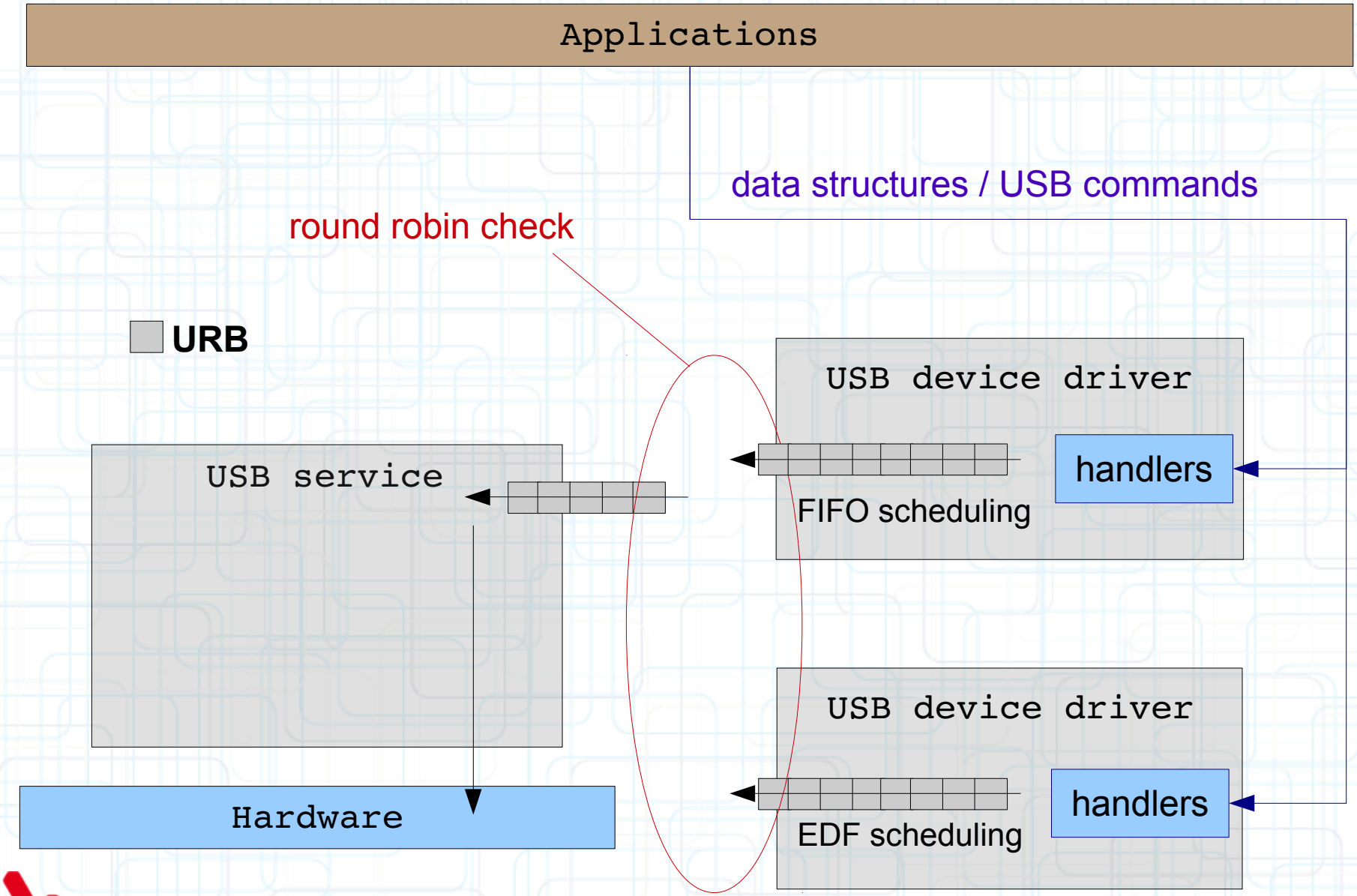
- Provide central USB service managing problematic fields
- Drivers handle *specific data*
- USB service handles *generic URBs*
- Access control dependent on use case
  - Use capabilities (see seL4) and extend them where needed
  - Use access control lists if suitable
- Inspired by previous paper and Helen OS

USB service: URB flow



Mierswa, Daniel ✉ daniel.b.mierswa@student.hs-rm.de
Tkocz, Daniel ✉ daniel.tkocz42@gmail.com

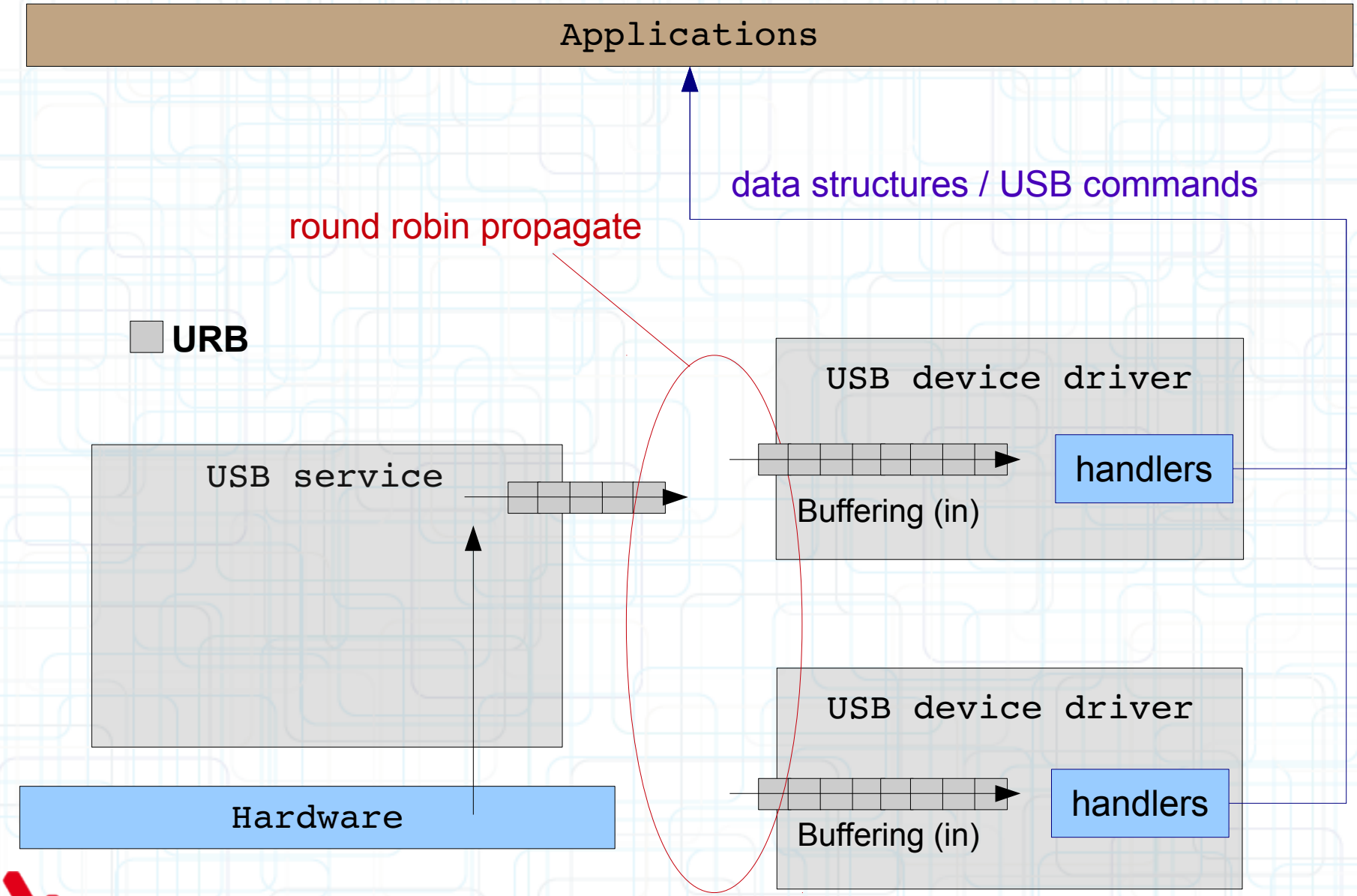
USB service scheduling (1)



Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim Geisenheim

Mierswa, Daniel ✉ daniel.b.mierswa@student.hs-rm.de
Tkocz, Daniel ✉ daniel.tkocz42@gmail.com

USB service scheduling (2)



Mierswa, Daniel ✉ daniel.b.mierswa@student.hs-rm.de
Tkocz, Daniel ✉ daniel.tkocz42@gmail.com

USB service access control

- Who is allowed to access drivers/devices?
- Drivers need capability from USB service to generate and queue URBs
- First application/client to open → Owner
- Owner can further restrict access (ACL?)
- Owner can be a library sitting on top of every application (trusted computing)

Mierswa, Daniel ✉ daniel.b.mierswa@student.hs-rm.de
Tkocz, Daniel ✉ daniel.tkocz42@gmail.com

Access restriction matrix (1)

	USB driver X	USB driver Y	USB application Z
USB command X	ALLOWED		
USB command Y		ALLOWED	
USB device Z			READ + WRITE

Mierswa, Daniel ✉ daniel.b.mierswa@student.hs-rm.de
Tkocz, Daniel ✉ daniel.tkocz42@gmail.com

Access restriction matrix (2)

	USB driver X	USB driver Y	USB application Z
USB command X	ALLOWED		
USB command Y		ALLOWED	
USB device Z			READ + WRITE

Capability bound to the domain
(used in USB service)

Mierswa, Daniel ✉ daniel.b.mierswa@student.hs-rm.de
Tkocz, Daniel ✉ daniel.tkocz42@gmail.com

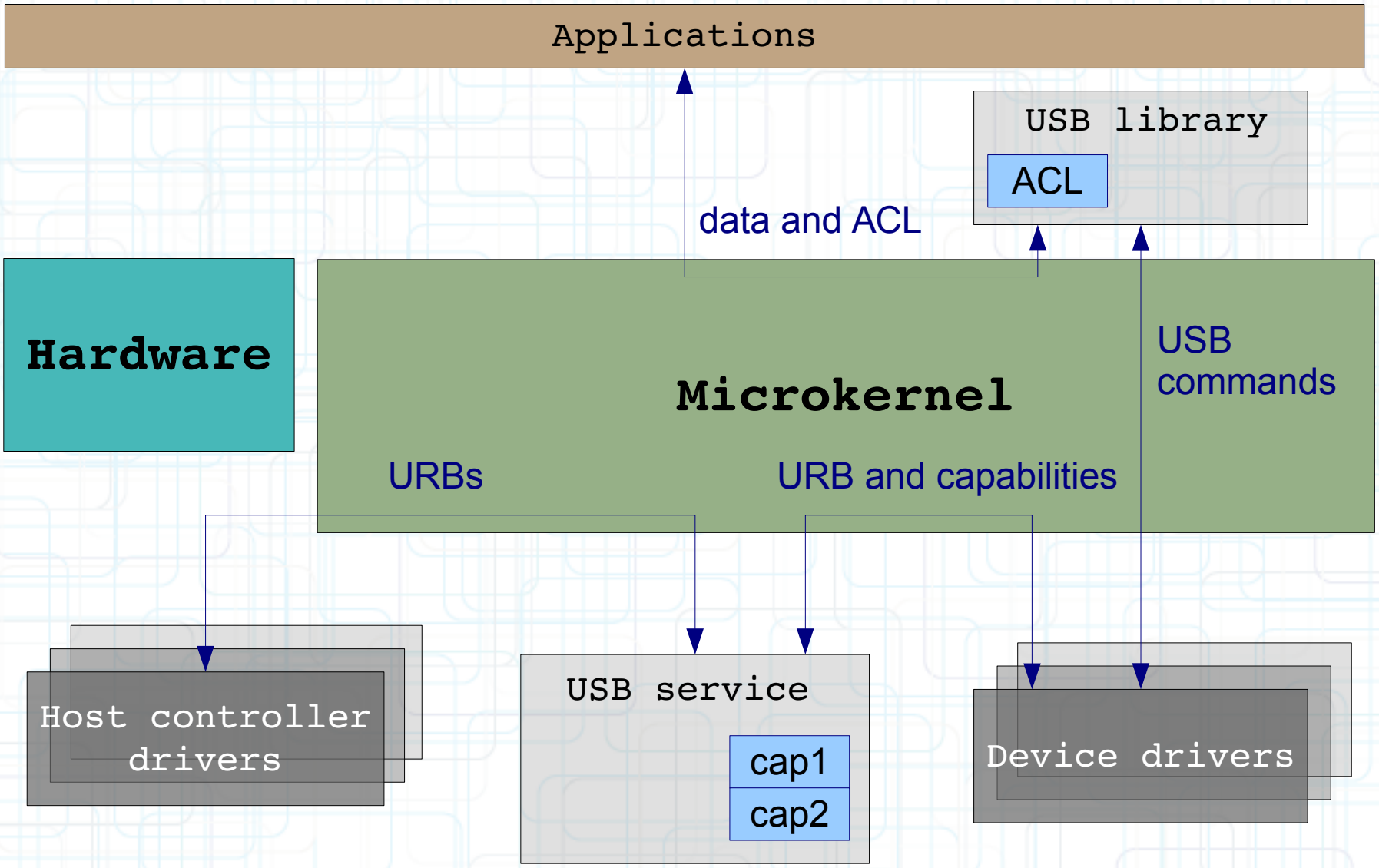
Access restriction matrix (3)

	USB driver X	USB driver Y	USB application Z
USB command X	ALLOWED		
USB command Y		ALLOWED	
USB device Z			READ + WRITE

ACL bound to the object
(used in USB library and drivers)

Mierswa, Daniel ✉ daniel.b.mierswa@student.hs-rm.de
Tkocz, Daniel ✉ daniel.tkocz42@gmail.com

USB service: final design



Mierswa, Daniel ✉ daniel.b.mierswa@student.hs-rm.de
Tkocz, Daniel ✉ daniel.tkocz42@gmail.com

Conclusions (1)

- Design ideas very theoretical
  - Next: implement prototype
  - Check how Linux drivers can be adjusted to fit in the design
- USB problems not very specific to the microkernel architecture
- Should've probably started work on paper earlier and implement prototype while writing (or earlier)

Mierswa, Daniel ✉ daniel.b.mierswa@student.hs-rm.de
Tkocz, Daniel ✉ daniel.tkocz42@gmail.com

Conclusions (2)

- Scheduling and access restrictions probably not real-life suitable
  - Should there be an owner at all?
  - URB Priority of HID probably higher than i.e. mass storage
  - Lots of indirection on first sight → probably problematic regarding performance

Mierswa, Daniel ✉ daniel.b.mierswa@student.hs-rm.de
Tkocz, Daniel ✉ daniel.tkocz42@gmail.com

Open discussion

Thank you for your attention!

Mierswa, Daniel ✉ daniel.b.mierswa@student.hs-rm.de
Tkocz, Daniel ✉ daniel.tkocz42@gmail.com